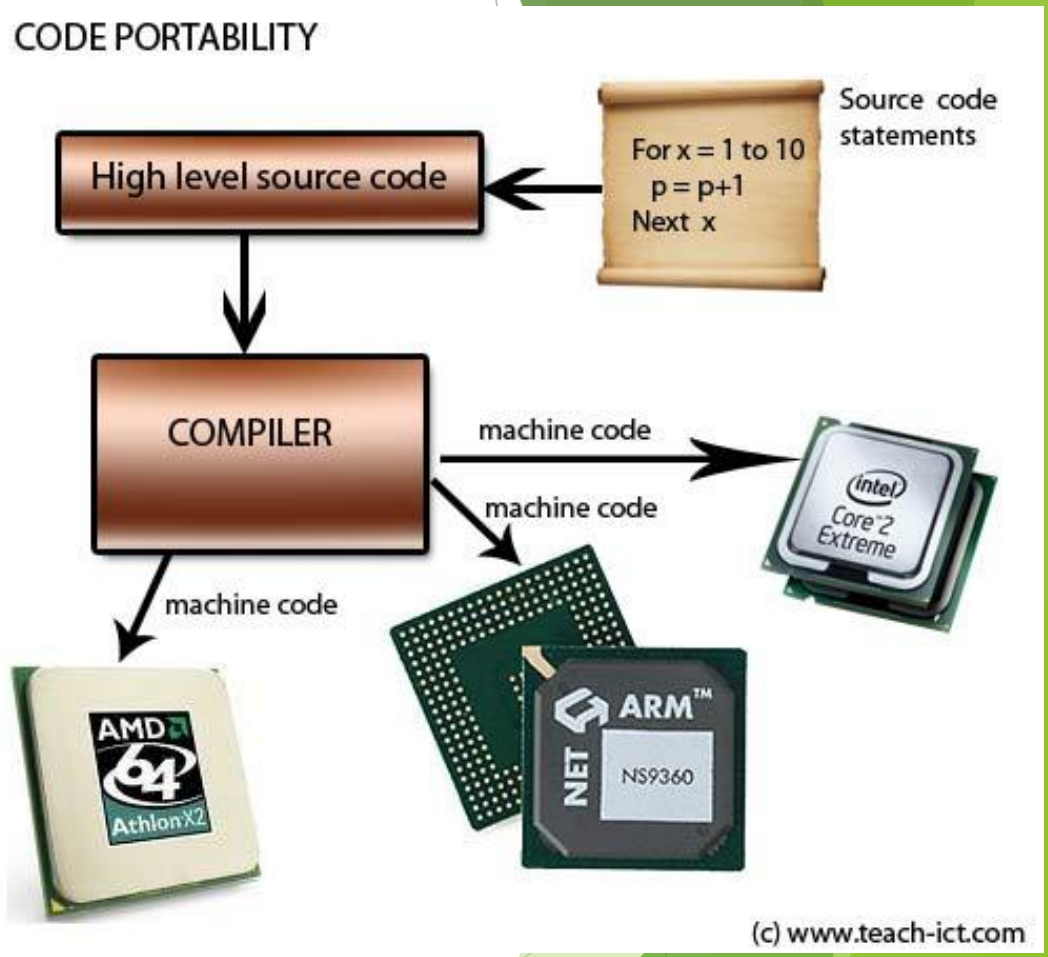
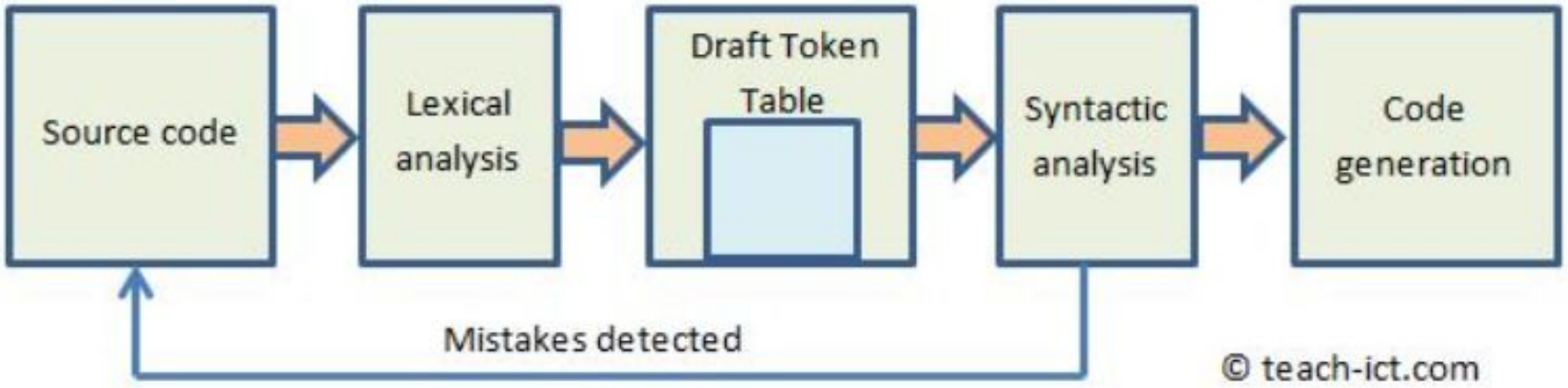
Compiler and Interpreter

A **compiler** is computer software that transforms computer code written in one programming language (the source language) into another programming language (the target language).





Lexical analysis is the extraction of individual words or lexemes from an input stream of symbols and passing corresponding tokens back to the parser.

Syntactic analysis, or parsing, is needed to determine if the series of tokens given are appropriate in a language - that is, whether or not the sentence has the right shape/form.

Code generation (object Code) is final phase of compilation. Through postcode generation, optimization process can be applied on the code, but that can be seen as a part of code generation phase itself.

The purpose of a 'compiler' is to translate source code into machine code.

A compiler translates the whole program as one complete unit

It creates an executable file

It is able to report on a number of errors in the code

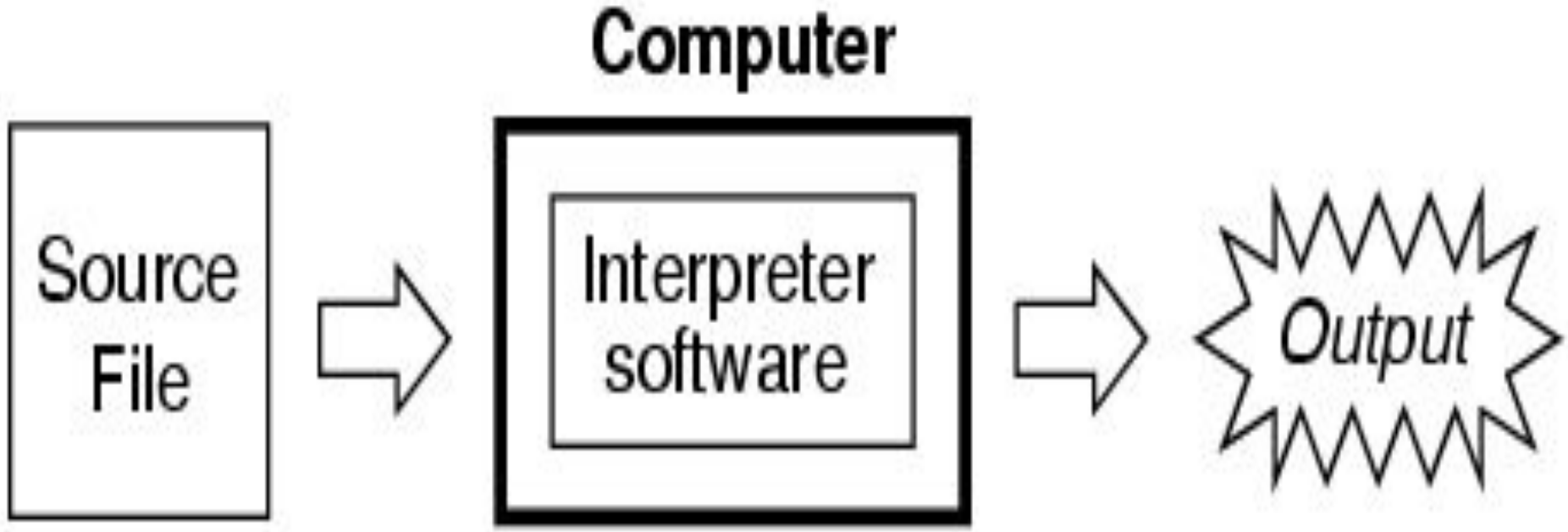
It may also report spurious errors

It does not need to be present in order to run the code

It can optimise source code to run as fast or as efficiently as possible

Code portability - a compiler can target different processors using the same source code

Interpreter is a computer program that is used to directly execute program instructions written using one of the many high-level programming languages

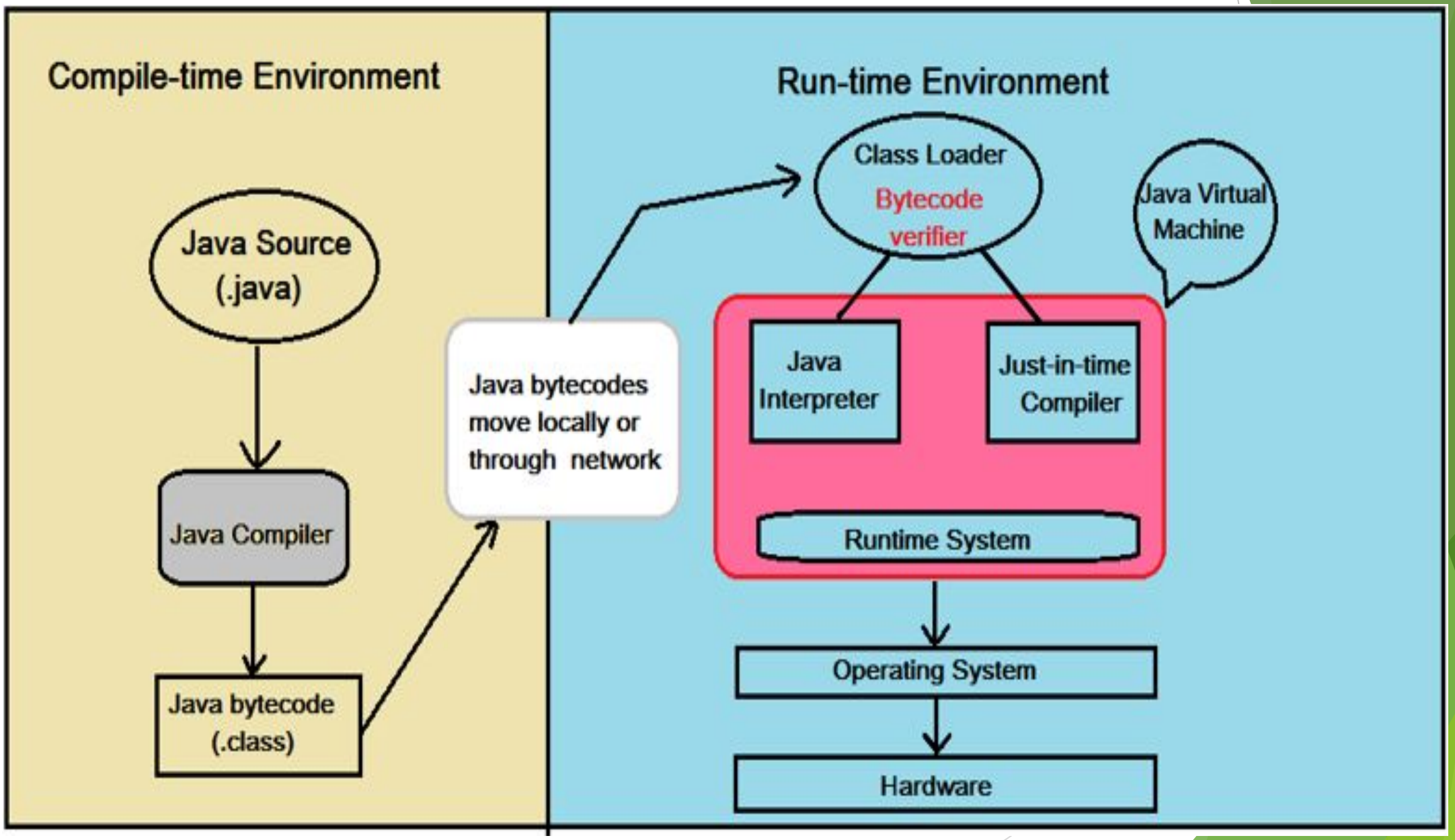


Summary

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

How Java is compiled & interpreted?

When you write a java program, the **javac compiler** converts your program into something called byte code. All the java programs run inside a jvm (this is the secret behind java being cross-platform language). Bytecode compiled by javac, enters into jvm memory and there it is interpreted by another program called java.
This java program **interprets** bytecode line-by-line and converts it into machine code to be run by the jvm. Following flowchart shows how a java program executes.


Compile-time Environment
Java Source
(.java)
Java Compiler
Java bytecode
(.class)
Java bytecodes
move locally or
through network
Run-time Environment
Class Loader
Bytecode
verifier
Java Virtual
Machine
Java
Interpreter
Just-in-time
Compiler
Runtime System
Operating System
Hardware